Go-Forward Plan

Smart Sleep Tracking Mattress for Children

Delivered to Serta Simmons Bedding in March 2019 by Thayer Group 14

Table of Contents

Embedded System

Firmware

The firmware for the Arduino is located in the file *sleepArduino.ino*. It contains a simple data processing algorithm to transform the three analog accelerometer readings to something understandable. Although the current data manipulation worked reasonably well to produce one number we could understand, we suggest SSB further investigate this processing by moving the accelerometer in defined motions with another, already calibrated, accelerometer to confirm readings are correct.

The firmware for the NodeMCU is located in the file *wifi_node.ino.* It contains logic to process state changes and transmit to the database one enter/exit/bed wet.

Both of these files can be found at the GitHub repository link https://github.com/rhmartin/sleep.

Setup

- **Current**: The device can connect to standard home WiFi and pair with the user account via an unsecured pairing process based on a unique device ID.
- **Recommendation**: SSB should add a custom serving page to the WiFi setup that collects the user's email & password and uses this information to sign the user in. This will be more secure and necessary once the database security issue is resolved & the system requires the Arduino to sign in via Firebase authentication (see database security recommendation). This process would also remove the insecure time period where a different user with the unique device ID could pair with the owner's device.
- See Set Up section below in Software for more information.

Updates

- **Current**: The updated firmware has to be manually re-flashed onto the controller via a USB cable.
- **Recommendation**: Wireless update capability should be added so that users can receive improvements.

Power Supply

- **Current**: The Arduino (5V) and NodeMCU (3.3V) each have their own power supply line via USB.
- **Recommendation**: Use a single 3.3V power converter to power each of the Arduino and NodeMCU.

Controller

- **Current**: Two individuals microprocessors are used in the project, the Arduino and NodeMCU.
- **Recommendation**: Use a single microprocessor to collect data and connect to the online database.

Controller Placement

- **Current**: The controllers are inside a box that is placed below the mattress springs in the top left corner. The power regulator is outside the bed at the wall plug.
- **Recommendation**: Conduct A/B tests with parents to see if they prefer the DIY step of placing the controller in the bed or if they prefer not having to do this. Also poll parents to see if they are comfortable having the controller in their child's bed. Possibly locate the box on the floor under the bed as the current Beautyrest Sleeptracker is configured.

Accelerometer

- **Current**: The ADXL355 accelerometer is calibrated and we delivered two locations, one on the side of the mattress and one in the mattress.
- **Recommendation**: Determine the optimal depth of the accelerometer in the mattress foam by further A/B testing; test an accelerometer that measures a smaller range with larger precision.

Load Cells

- **Current**: The two load cells are placed on a piece of plywood under the mattress and provide coverage for 93% of the bed.
- **Recommendation**: Investigate the benefit of adding a third load cell to cover more of the bed so the child is detected when on the very edge.

Conductive Thread

- **Current**: Stainless Steel Conductive thread is sewn in to a mattress topper with five inches of gap between the threads. The thread takes on the pattern shown in Figure 1.1. The thread was sewn so that the two threads never cross and each maintains its connection (threads tied together when a roll ends). A layer of cotton fabric was sewn over each thread to prevent wire crossing. The fabric stitching never crossed over the conductive thread. Further design decisions are detailed in Appendix 1.

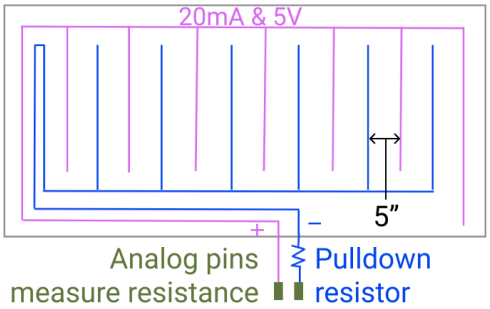


Figure 1.1: Conductive Thread Design

Recommendation: The layer of fabric over the thread effectively covered the conductive thread from sight and made the circuit resistant to shorts from crumpling the topper. We replicating this method, although the covering material does not need to follow the conductive thread so closely, as long as it covers it.

In Figure 1.2, a slight ‘H’ pattern can be discerned from the spillage. It was found that the capillary action of the cotton cover is stronger than the capillary action of the mattress topper. Thus, if water is poured directly on a line, it is more inclined to follow the line in a H or T pattern than to seep over to the other line and complete the circuit. This was not a major issue due to the close proximity of the lines.

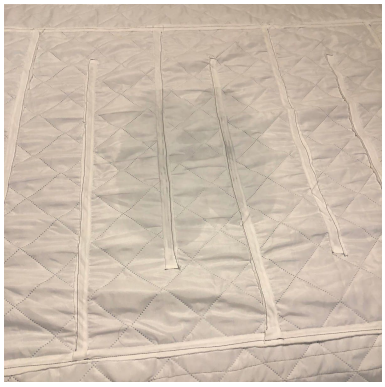


Figure 1.2: Wet Mattress Topper Prototype

Graphical User Interface

Moving forward, we recommend that Serta Simmons make several changes to the logic of our sleep algorithms and app, data visualization, and database structure. At the highest level we have generally implemented the front-end, but have not created any logic on the server side. This is the most pressing addition. The server should include logic for better processing data from the arduino and serve specific data to the app rather than all at once.

System Setup

Current:

The app allows users to sign up, pair to their device, and calibrate their device. The device can be connected to any standard home WiFi (with WPA or WPA2 security)* and paired with a user account via a unique device ID (DUID) that should be both stored in the embedded system and printed on the product. The installation procedure is as follows:

1. Place mattress on bed
2. Plug the mattress into the wall
3. Download the app
4. Create a new account on the app (Instructions after this point are also displayed on the app)
5. Connect the device to WiFi:
    a. Go to your phone's settings and connect to "ESP\#\#\#\#\#"
    b. After connecting and being redirected to a sign-in page, click "Configure with Scan"
    c. Input your network name (SSID) and password
    d. Wait for the screen to close (WiFi connection will automatically return to your standard settings and network)
    e. Return to the app
6. Pair device with user account:
    a. Click 'Add Tracker' in the app
    b. Input and submit the unique device identification number (written on the product and instructions).
    c. Await confirmation
7. Calibrate the device:
    a. Click 'Calibrate Device'
    b. Have subject sit on bed and press 'Calibrate'
    c. Wait for confirmation (~15 seconds)

8. Device is now paired and calibrated. Calibration must be repeated if device loses power. Data will now populate for new events (App may need to be refreshed).

This procedure has security flaws. If in possession of someone's DUID, an unauthorized user could pair with their device while it is in the setup phase. See below for backend description of this process, under Database: Current Structure.

> *The embedded system utilizes the package: Avoid Hard-Coding WiFi Credentials on Your ESP8266 Using the WiFiManager Library
> (https://www.instructables.com/id/Avoid-Hard-Coding-WiFi-Credentials-on-Your-ESP8266/)

Recommendations: To address the security flaw in this process, and as mentioned below in Database: Security, this procedure should be modified so that the user inputs their network information and their account information in the local WiFi network served by the embedded system. This would likely require either editing the existing package (above) to pass more information & serve a new webpage form or utilizing a different WiFi serving & connection package.

Database

Current Structure

The database currently contains two concurrent structures. The old hierarchy was structured for a single user and simply contained each night with the timestamps of sleep events as a sub-hierarchy. The current hierarchy contains user profiles, nightly sleep data under 'userData', and pairing infrastructure under 'Pairing'. 'userData' contains an object for each user, keyed by the unique user id (UID) created by Firebase authentication. Under each user, the structure mirrors the structure of the old hierarchy. This more recent hierarchy is:

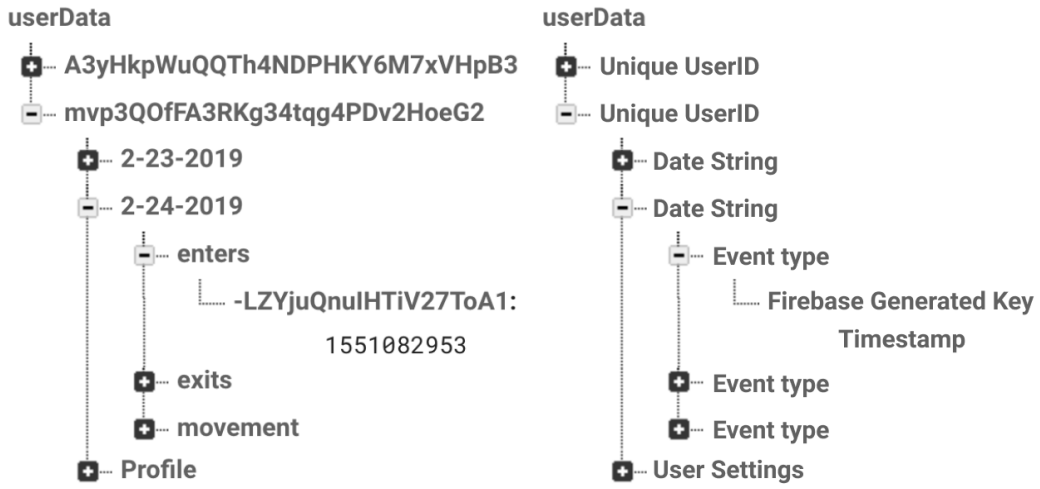


Figure 2.1: Database Hierarchy

The generic file paths through the database has two branches outlined here:

```
./sleep-Data/
     userData/[uniqueUID]/
          [ String "mm-dd-yyyy"]/[event type]/[timestamps]
          [uniqueUID]/Profile/[various settings]/[0 or 1]
     Pairing/[uniqueDUID]/{UID: UID or 0}
```

The Pairing part of the database stores DUID for each device that has been powered on and connected to WiFi or any user account that has started the 'Add Tracker' process. It stores either '0' or a UID depending on whether the device has added it or the user has submitted the DUID in the app. Upon completion of the pairing process, the user's 'Profile/hasD' will be updated to 'true'.

Read/Write

- **Current**: Direct write and read by the hardware and app, app pull all of the users data every time.
- **Recommendation Short Term**: Downloads must be minimized to minimize cost, the current set up is projected to cost ~$50/user/yr while an optimized database may only cost $4.20, see cost projections below. This optimization likely requires setting up functions for pull requests (likely via HTTP) that only send the most recent day, store old data locally on the device. Data appears to be around 0.35 MB/day so this should not be a burden to the device while downloads have averaged around 150 MB/day, see cost projections below. This will likely require saving days as ```Date Object``` keys instead of strings so that the days can be sorted easily. It may also require saving the most recent day downloaded in the application.
    - If local storage is not desired, data should only be pulled as it is accessed (ie. only pull all data if the user clicks on the "All" view).
    - See this article as a warning of what can happen: [https://pamartinezandres.com/lessons-learnt-the-hard-way-using-firebase-realtime-database-c609b52b9afb]
- **Recommendation Long Term**: Ultimately another database may be more economical, especially if optimization is not completed

Cost Projections

Firebase charges for the number of GB stored as well as the number of downloads. Downloads are currently very high as discussed above, which drives almost all of the costs.

The formula is the sum of the cost of storing data as it accrues and the average downloads per month.

$/user yr = (stored MB/mo.)*(sum(1-12))*($0.005/MB) + (downloads MB/mo.)*($0.001/MB)

Data Structure	Data/mo. (MB)	Download/mo. (MB)	User/yr ($)
Current	9.9	4590	59.10
Optimized	9.9	24.9	4.20

Table 2.1: Table of Costs per User of Firebase Database given Current App and Potential Cost for a Download-Optimized Application

Security

- **Current**: Read/write rules in Firebase are set according to the Firebase docs to only allow authorized users to access data under their UID. It does not.
- **Issue**: The front end requires authentication but the backend can still read and write without authentication.
- **Recommendation**: Check rules, test protection, write new rules, seek other authentication or database options.

The database also still contains our old data that is not sorted under a userID. This has not been deleted as it may be of use in displaying example data if sign in is disabled or as example to an unregistered user.

General

- **Timestamps**: Finalize as 10 or 13 digits (with or without milliseconds in Epoch time) and adjust app logic accordingly. Currently timestamp digits are 10 for events and 13 for movements.

Logic/Algorithms

Improve Sleep Algorithm

- **Current**: the app in `HomeScreen.js` in `onFetchData()` executes a simple algorithm that checks that the child's time in bed is greater than a threshold amount, and that the child's restlessness in the five minutes after the "asleep" period begins is less than a 30 out of 100.
- **Current**: the napping algorithm (also in `onFetchData()`) categorizes any "asleep" period as a nap if it is less than 4 hours in duration and between the hours of 10 a.m. and 6 p.m.
- **Recommendation**: Further testing into thresholds or *rewrite algorithm completely after larger scale data collection and possibly apply machine learning after testing with a sleep lab or other definitive method.* From our experience, machine learning algorithms such as decision trees, regressions, naive Bayes, nearest neighbor, support vector machines, and deep neural networks may be useful.
    - After categorizing awake vs asleep, identify as normal sleep, restless sleep, or nap.
    - Do not report light/deep/REM sleep as this is too complex to be useful for parents and can be misleading (according to DHMC sleep doctor).

Multiple Children

- **Current**: Settings include button to add child but no backend.
- **Recommendation**: Add a page for parents to input their child's information and add a second tracker to their account.
- **Related**: Add navigation between children, database structure for support.

National Averages

- **Current**: National averages are hard-coded in (only the average hrs of sleep is accurate). Opt in/out settings exist for national data pooling (front end and database).
- **Recommendation**: Once SSB has gathered enough data or can access data from existing customers, calculate these averages and split based off of age group demographics.

Multiple Users

- **Current**: The app and database support multiple users using Firebase's email & password method, user data stored under UID (unique user id - automatically generated by Firebase), and read/write rules specified in the database to restrict to authorized and matching users.
- **Issue**: The hardware is able to write to the database without signing in if it has the UID hard coded in, meaning the Firebase rules are not effective. Again, the rules do deny read permission from the app side if not signed in, so this seems to impact writing from the arduino but not reading from the react native application.
- **Recommendation**: Implement other sign on methods, *server logic for creating initial user branch of database via the server not the app.*

Data Processing/Usage:

- **Current**: The app downloads the entire database, processes it from the JSON object of strings into js arrays, analyzes data with algorithms described above. The app then stores multiple arrays in state including every day of data and splits of data for each view. This creates some lag in the application and extraordinarily high download amounts (See section below on Database improvements).
- **Recommendation**: Store more useful data in the database, possibly performing all logic on the database and only sending data to be directly used in graphs and data summaries, fetch only relevant data rather than always the whole history, store less movement/restlessness data.

UI

Tutorial

- **Current**: `i` button toggles explanatory/tutorial style text on views, user can read to walk through app functionality, each title also is clickable to bring up an alert about what the metric is.
- **Recommendation**: Add interactive introductory tutorial walk-through for the first time the user opens the app.

Customize Metric Displays

- **Current**: Displays the same metrics regardless of preference, except for main week and month graphs which can be toggled from total hours broken down by awake/asleep/napping and time into and out of bed. Displays trends of data on Month page, details on day page.
- **Recommendation**: Allow parents to customize the data metrics shown on the summary page, e.g. show bedwet or don't, and/or include more averages such as average bedtime or waking time.

National Averages/Recommendations

- **Current**: Hardcoded, fake national averages, with recommended line for sleep amount.
- **Recommendation**: Add recommendations specific to their child about how much sleep to get, how many exits/wets are normal each night, how restless their child should be.
    - Consult with medical professionals and/or determine "normals" based off testing results
    - Clarify movement (on a 0-100 scale) by telling parents what a goal range should be, e.g. Are most children of the same demographic scoring a 30-40? 80?

Movement Presentation

- **Current**: Line graph from low to high of movement. This scale is from 0-100.
- **Recommendation**: Clarify range, include some sort of baseline. We tested:
    - Averages on the scale of 0-1, 0-10, and 0-100, but the range seems to make no difference in user confusion
    - The descriptions (low, normal, high) also confuse people, without them people are even more confused
    - We suggest dedicating a portion of the tutorial to explain to users what the movement metrics mean

Clicking Bar Chart

- **Current**: Click on bar to update button above graph to show summary of that day, click button to go to day detail.
- **Recommendation**: Pop up a small, hovering box detailing important sleep details for that day.
    - By clicking on that box, parents could then navigate to the daily view
    - This will clean up the user interface and make it more intuitive

General Layout

- **Current**: Flat, little busy, no delimitation between sections, see screenshots above.
- **Recommendation**: Align visual layout more closely with existing SSB sleep tracking apps, slight background shading behind graph to give depth/section it off or possibly borders between sections to visually break up and compartmentalize the main views .

Descriptions of MVP Graphical User Interface Components

Firebase Database

The Google Firebase realtime database that our app and sensor suite interfaces with is: https://sleepdata-d5465.firebaseio.com/

The database currently contains two concurrent structures. The hierarchy being phased out was structured for a single user and simply contained each night with the timestamps of sleep events under each night. The new hierarchy is under ‘userData’ and contains an object for each user, keyed by the unique user id (UID) created by Firebase authentication. Under each user, the structure mirrors the structure of the old hierarchy. The first level contains dates for each night of measured sleep. Under each date, the second hierarchy contains time-stamped array of sleep events. Each "enters" contains an array of Epoch timestamps for each bed enter, "exits" contains an array of Epoch timestamps for each bed exit, "movement" contains a string array of Epoch timestamps and corresponding restlessness averages (averages rolled up every threshold number of seconds as determined on the Arduino), and "wets" contains a string array of any bedwetting events. Although the app can handle missing "exits", "movement", and "wets" arrays, a missing or empty "enters" array will cause the night's data to be disregarded.

Mobile App and Code

The Thayer team has delivered a GitHub repository link to all project code, a mobile app published on Apple TestFlight, administrative access to a Google Firebase database containing test data, and this Go-Forward Plan to SSB. Our app is programmed in React Native, uses Expo, interfaces with Google Firebase, and visualizes graphs with the Formidable Labs Victory Native package. The code is well-commented, but in summary it contains 13 main files:

`AllScreen.js`
This file is located in the components folder and is a component of `HomeScreen.js`. It contains the rendering code for the "All" data view page. In this view, a full data report of sleep is reported in a stacked bar graph showing the number of hours asleep, awake, and napping for each day. By clicking on a bar, the user can switch the daily details displayed under the "Sleep" title. By clicking on those daily details, the user is transported to the daily view, as rendered in `DayScreen.js`. Below the sleep graph, line graphs showing data trends are shown for movement, bed exits, and bedwetting. The movement chart graphs the daily movement average, ranked on a scale of 0 to 100. The bed exits graph shows the daily amount of times the user exited the bed each night over time, and the bedwetting graph shows the daily number of bedwets. Finally, averages for each night's sleep duration, movement, bed exits, and bedwetting events is provided for the full data range.

`AveragesScreen.js`

This file is located in the components folder. It contains the rendering code for the national averages comparison, accessible by the white graph button on the right header of the app. Bar graphs showing the child's all-time averages compared to the national averages for their age demographic are shown for sleep duration, movement, nightly bed exits, and nightly bedwetting events. On the sleep duration chart, a line showing the recommended hours of sleep for the child is also drawn. Currently, the national averages and recommended sleep duration are hard-coded in. Moving forward, SSB should calculate these national averages from the data of all their users and consult with sleep doctors to determine the recommended sleep for each unique child.

`CalibrateScreen.js`

This file is located in the components folder. It is navigated to through the "Calibrate Tracker" button on the Settings page. This page displays instructions, a calibrate button, and will provide confirmation when calibration is completed. The process is: a child should sit on the edge of the bed and press the calibrate button. This writes a "1" to the 'Profile/calibrate' variable in the Firebase database. When this equals 1, the sensor suite records the load cell readings over ~10 seconds to create a reference of how much the child and mattress weigh. After saving this reading, the sensor suite sets the calibrate variable to 2 in the database signifying completion. Anytime the variable is 0, the app will prompt the user to calibrate the device.

`colors.js`

This file is located under the components folder. It contains the color variables for all app graphics. It is used to facilitate changing colors on the style sheet and Victory Graphs. The graphs cannot receive a style component from the stylesheet so colors are hardcoded into each graph, the color object is used to keep these in one easy to edit place.

`DayScreen.js`

This file is located under the components folder and is a component of `HomeScreen.js`. It contains the rendering code for the "Day" data view page. Before rendering the graphics, the file computes two arrays for the sleep graph, `ySleep` and `in_out`. `ySleep` contains the timestamps of each enter or exit for the night. `in_out` contains a 0 if the user left the bed at the corresponding time in `ySleep`, or a 1 if the user entered the bed at the time. Both of these arrays are pushed to sleepData in order to be viewed graphically. In terms of graphics, the page renders a sleep graph displaying the total duration of sleep along with sleep times. Times where the user was in bed asleep are shaded blue, and times whee the user was out of bed or not asleep are not shaded. Underneath, a line graph showing all movement readings over the night is drawn along with a movement average on a 0-100 scale. The number and times of bedwets and bed exits are also included.

`HomeScreen.js`

This file is located under the components folder. It contains the main processing and navigation code for our app. It also is the main file interfacing with the Google Firebase database. There are three main functions. The first, `toggleTutorial()` allows the user to turn tutorial mode on and off by pressing the white i button on the left header. When tutorial is toggled on, white text appears on each screen walking the user through the different app functions. The second, `toggleABtest()`, allows the user to toggle between sleep graphs on the "Week" and "Month" views by pressing the views button on the left header. The default view shows a graph of the sleep durations for each night. The second view shows a sleep graph with timestamped bedtimes and waketimes for each night. The final and most important function, `fetchData()`, references the `onFetchData` variable.

This variable is a snapshot of the Firebase database. First, the function scans through the Firebase hierarchy, only reading sleep data from objects labeled as dates in string form "m-d-yyyy". It reformats the string data as a date object, and fills in empty data for days that are missing from the database. Next, it maps the enters, exits, wets, and movement arrays from the database into corresponding arrays in React Native. In the middle of our project, we encountered an error with sending timestamps in 10 digit vs 13 digit Epoch time. To catch this error, we append three 0s to all 10 digit timestamps that are read in. Effectively, this is setting the milliseconds of the timestamp to 0 if they do not already exist.

For our data cleaning, we check that enters and exits exist, and that there are the correct number of corresponding enters and exits. If not enough enters exist, we fill a fake enter halfway between the two consecutive exits. If not enough exits exist, we fill a fake exit halfway between the two consecutive enters. If the child is still in bed, we display their sleep data through the current time. We calculate sleep times and durations by looking at the difference between consecutive exits and enters. If the time difference is greater than a hard-coded `asleepThreshold`, and if the average restlessness in the five minutes after entering bed is lower than a hard-coded `restlessThresh`, we consider the user asleep. If the sleep period is less than four hours in length and between the hours of 10 am and 6 pm, we categorize the sleep as napping. Awake, asleep, and napping times and durations are used in our data visualizations in other components.

All data for each night is pushed to the `nightData` array and eventually saved in state as boards. Each board contains the `day` (string date from Firebase), `dateLabel` (mm-dd string date), `exited` (array of exit timestamps), `enters` (array of enter timestamps), `bedwet` (array of bedwet timestamps), `sleep` (number of hours of sleep for the night), `restTime` (array of timestamps corresponding to each restNum), `restNum` (array of restlessness averages throughout the night), `inBed` (total time in bed from first enter to last exit), `dayLabel` (label associated with the day of the week Sun-Sat), `awake` (total time between first

enter and last exit where the child is awake), `naps` (total time napping), `restlessAvg` (average restlessness per night).

In addition to boards, other state data includes `displayBoards` (only the boards displayed by the Week component), `monthBoards` (only the boards displayed by the month component), `dateDic` (a dictionary of all dates from Firebase), `picked` (an index showing which board is being displayed by the day component), `firstRun` (a variable used in setting our default views upon opening the app), and `isLoading` (a boolean to render graphics after data is loaded).

We also calculate averages, convert timestamps to readable formats, allow the user to scan through days/weeks/months, and execute other functionality in the file. At the end of the file, we set up the functionality to toggle between day, week, month, and all data views.

Homescreen also checks the calibration status, prompting the user to calibrate when `[UID]/Profile/calibrate: 0`.

`MonthScreen.js`
This file is located under the components folder and is a component of `HomeScreen.js`. It contains the rendering code for the "Month" data view page. Before rendering the graphics, the file sets up `offsetData`, the array containing information for the second toggleable view of the sleep graph. This array contains information about bedtimes, waketimes, and labels for the second toggleable graph.

Depending on the view toggle state, the file first draws a bar chart showing either total sleep duration for each night or the bedtimes/waketimes for each night. Like in `AllScreen.js`, clicking on bars and bar details allows the user to view daily sleep details or the day page. Underneath the sleep chart, monthly averages for sleep duration, movement, bedwets per night, and bed exits per night are displayed.

`PairDevice.js`
This file collects the unique device id (DUID) and adds the signed in user's UID to the database under `./Pairing/[DUID]/UID: [UID]`. The file confirms this pairing when `./userData/[UID]/Profile/hasD` is set to true. The page displays a confirmation screen with multiple options, but the alert from homescreen regarding calibration will trigger on top of this page. This is a bug that should be worked out to allow the user to proceed to calibration if they choose.

`SettingsScreen.js`
This file is located under the components folder. It contains the rendering code for the settings page and is accessible by clicking the settings gear on the left header. Currently, the settings

screen displays the current user, allows the parent to calibrate their tracker, and enables customizable notification settings for various sleep events. At the bottom of the page, a "Save Settings" button allows the user to change their notification preferences stored in the Firebase database. "Log out" allows them to log out of their account. Going forward, the notification settings saved in Firebase should be used to customize when notifications are sent to the user. A privacy agreement, child details screen, and method of adding children to the parent's account should also be implemented.

`SignIn.js`
This file is located under the components folder. It contains the logic and rendering code for the sign-in page and multiple user functionality. User accounts are username and password protected in order to address privacy concerns.

`style.js`
This file is located under the components folder. It contains the different style variables used throughout the app.

`WeekScreen.js`
This file is located under the components folder and is a component of `HomeScreen.js`. It contains the rendering code for the "Week" data view page. Before rendering the graphics, the file computes an `offsetData` array similar to that in month which contains the processed data for the second toggleable view of the sleep graph. In terms of graphics, this file renders a graph showing either sleep duration or bedtimes/waketimes for each night of the week. Below, it displays weekly averages for sleep duration, movement, bedwets per night, and bed exits per night.

`App.js`
This file contains the root stack for our app. It also sets up the beginning notification-handling structure. Currently, the app prompts users to register for push notifications with Expo. Using Expo's test push notification server, the developers can manually send notifications to specific users. However, this functionality has not been fully built out with the app on Apple TestFlight. In addition, we have not programmed any back-end logic for sending user notifications or referring to user notification preferences.

`Firebase.js`
This file contains the Firebase database connection information for our app. By changing this file, SSB can change which database the app interfaces with. All configuration ID details are easily found on Firebase.

Appendix 1: Embedded System Design

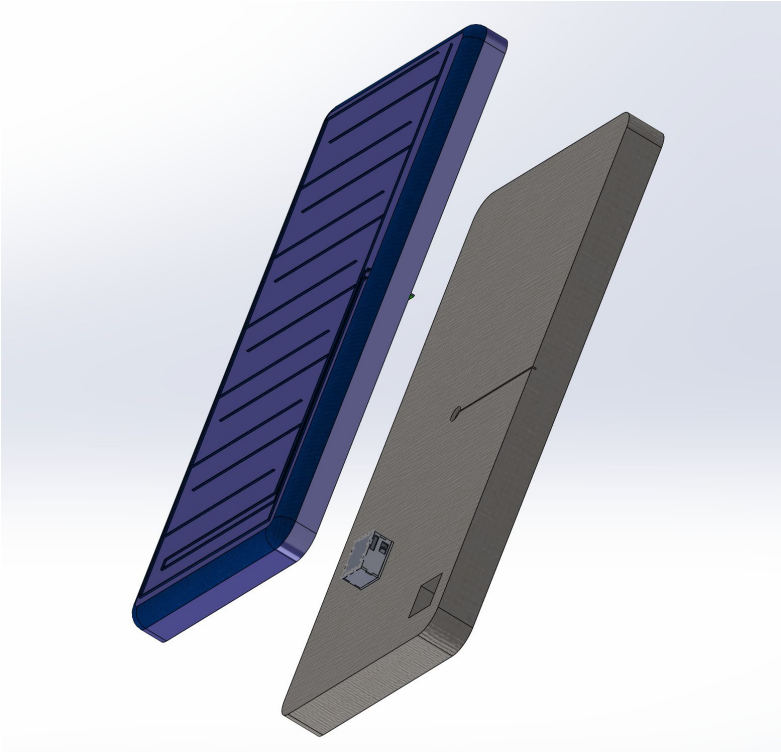


Figure A.1.1: Exploded View of Mattress Layers from Above. The electronics box can be seen floating between the layers. The indent of the accelerometer and its wire can be seen on the bottom layer and the conductive thread can be seen from above.

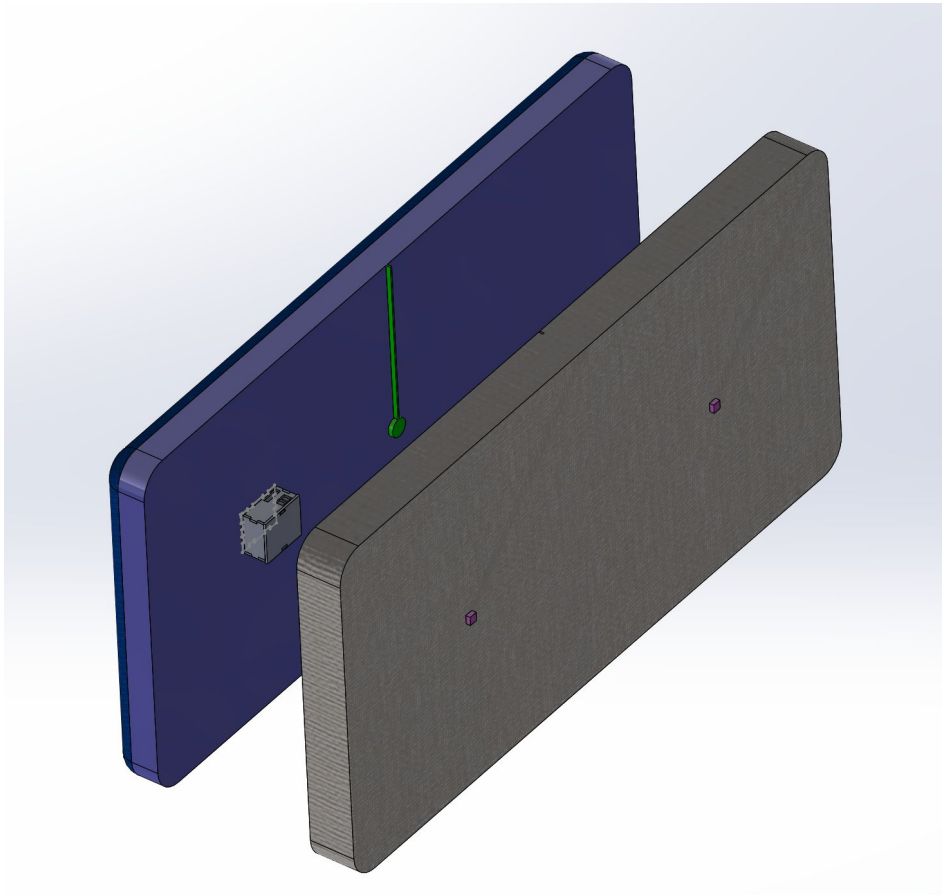


Figure A.1.2: Exploded View of Mattress Layers from the Bottom. The accelerometer and its wire can be seen between the layers, as well as the electronics box. The load cells can be seen as pink squares from below.

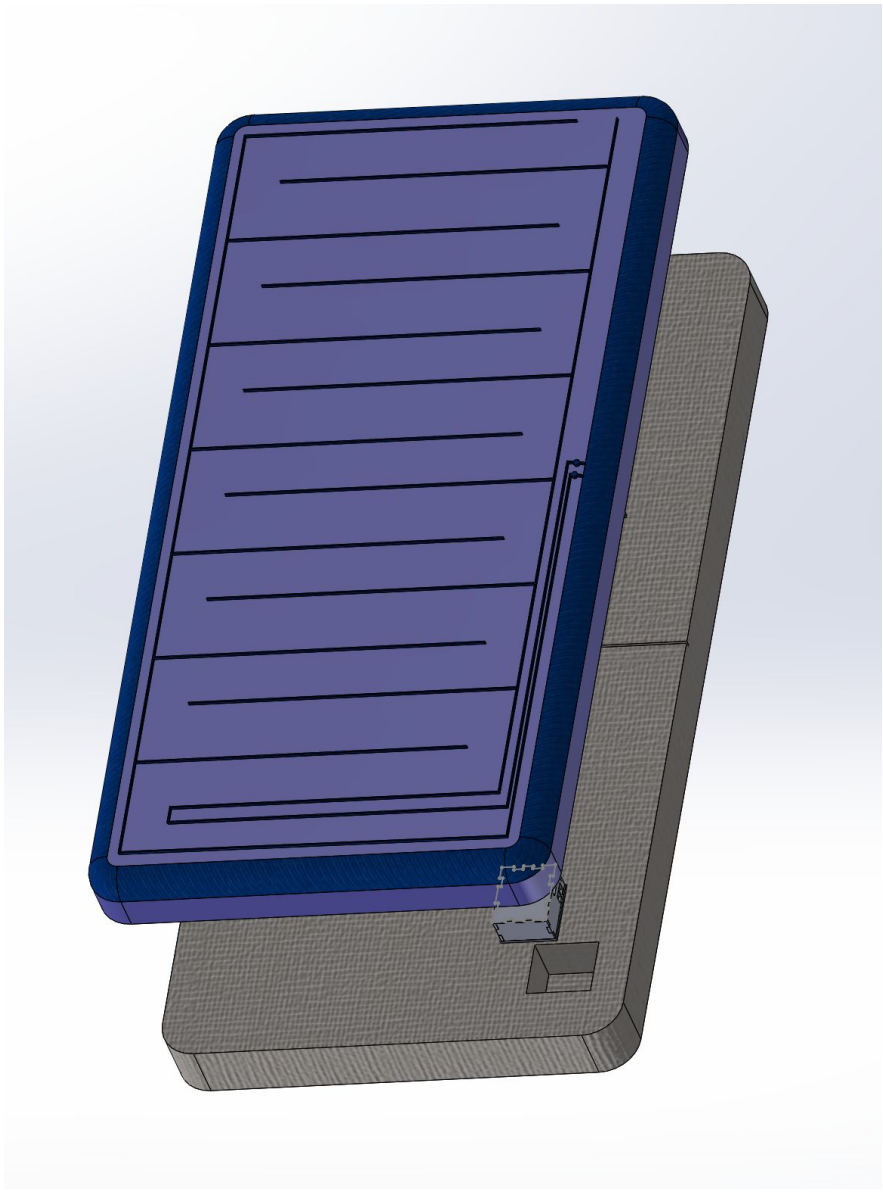


Figure A.1.3: Exploded View of Mattress Layers. The box for the electronics can be seen elevated from its cross section in the spring layer of the mattress. The conductive thread can be seen on the top layer.

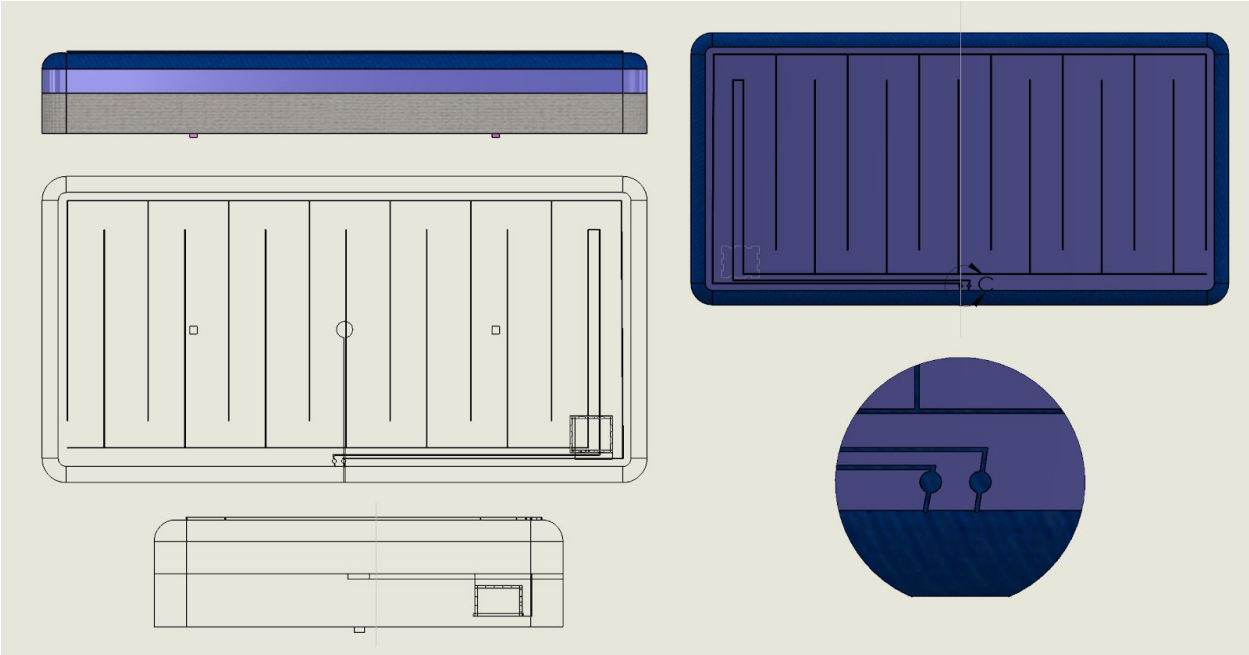


Figure A.1.4: Schematic of Mattress Design with Embedded Hardware. The right mattress has circle 'C' giving a close-up view of the conductive thread termination. The left mattress image gives a transparent view of the mattress from the top with the load cells, conductive thread, accelerometer, and box.

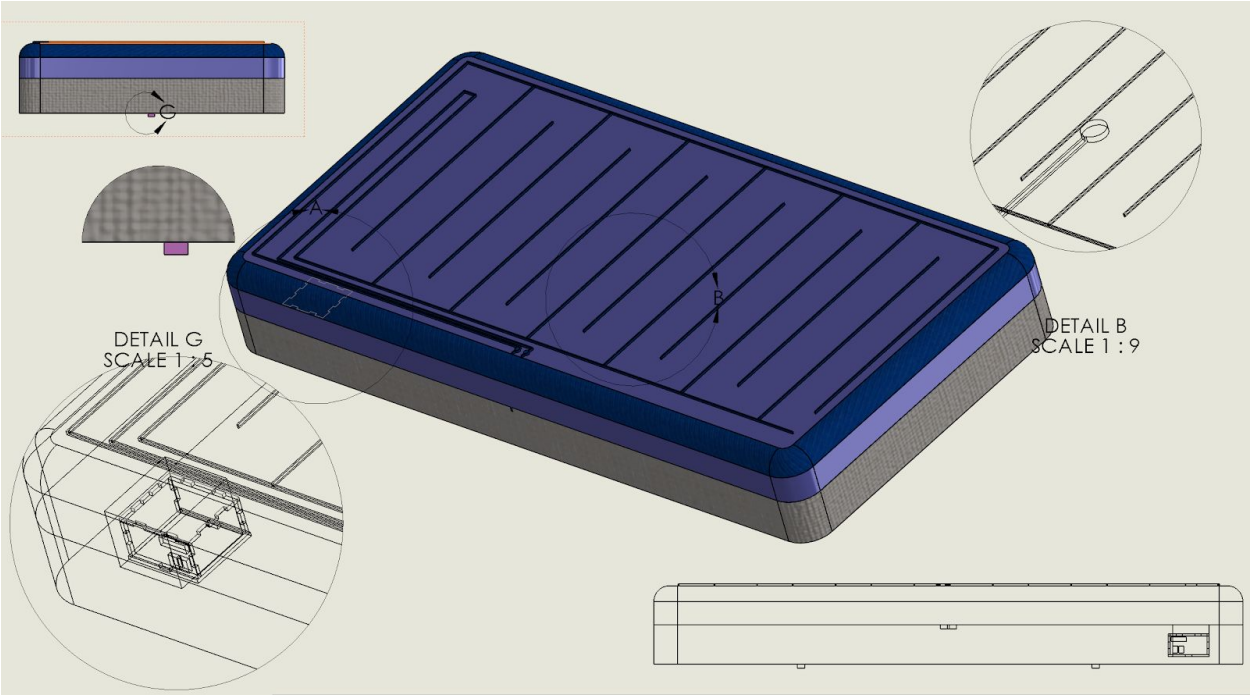


Figure A.1.5 Schematic of Mattress with Electronics Embedded. Circle A shows a close-up of where the electronics box is located inside the mattress. Circle B shows a close-up of where the accelerometer is inside the mattress. Circle C shows a cross-section of where the load cells go under the mattress. The middle shows the conductive thread pattern on the full mattress.

Electrical Hardware Overview

An electrical hardware overview including system block diagrams, component details, PCB layout, hardware costs, and datasheets is delivered to SSB in addition to this document.

Electrical Hardware Components

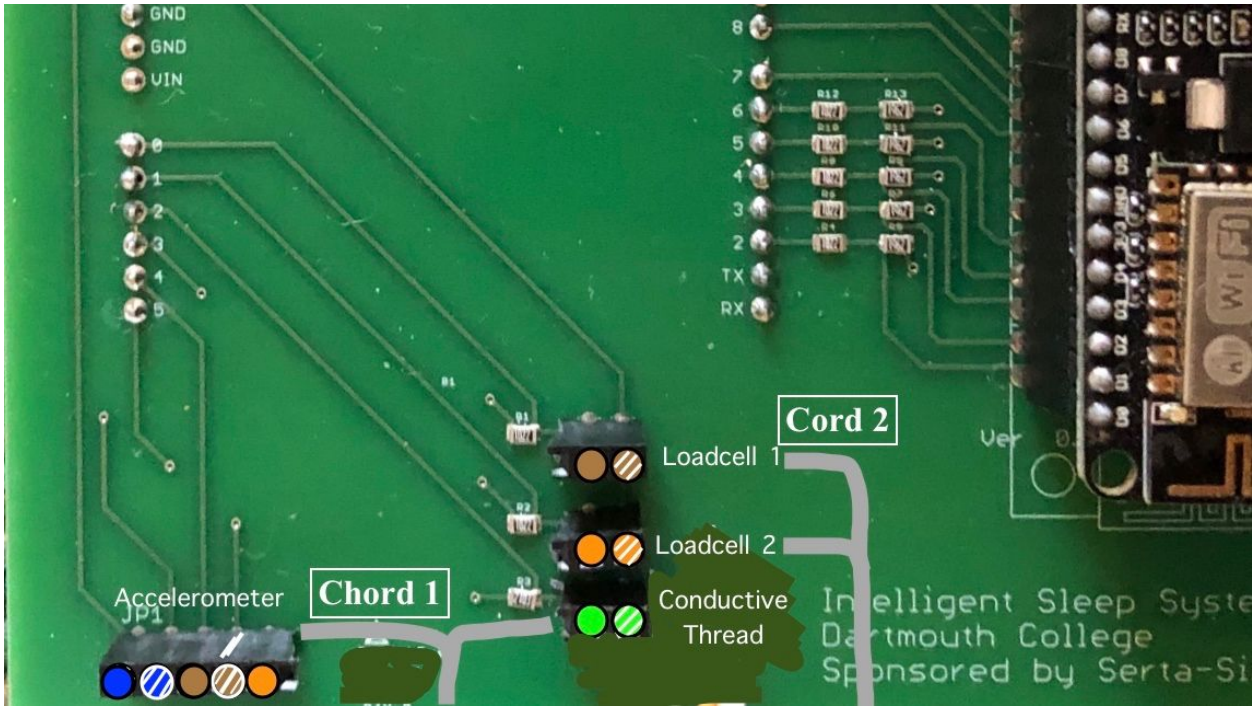


Figure A.1.6: PCB Connections. Schematic showing the wire colors from each cord that go into the different pins and their associated sensor.

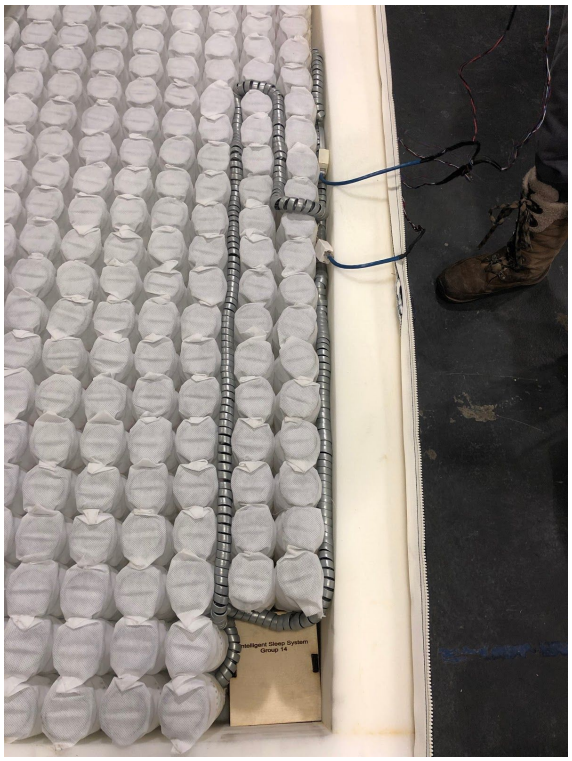


Figure A.1.7: Controller Installed in Mattress Coils

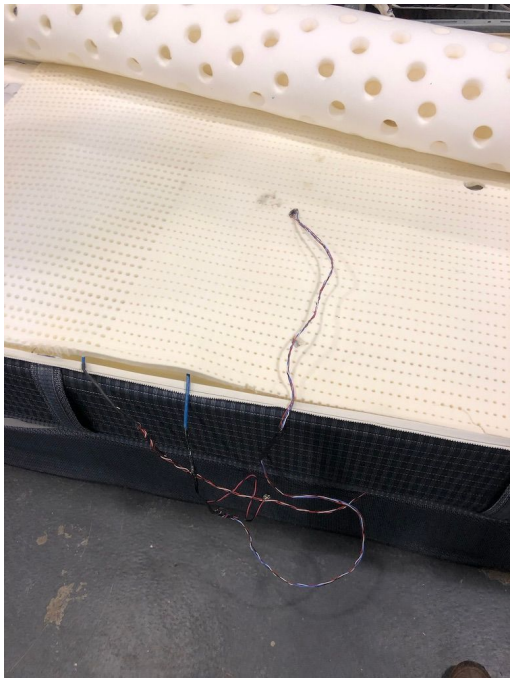


Figure A.1.8: Accelerometer Placed In Mattress Under Top Layer of Foam

Figure A.1.9: Termination of Conductive Thread to Snaps at Edge of Bed (without snaps attached or cover)

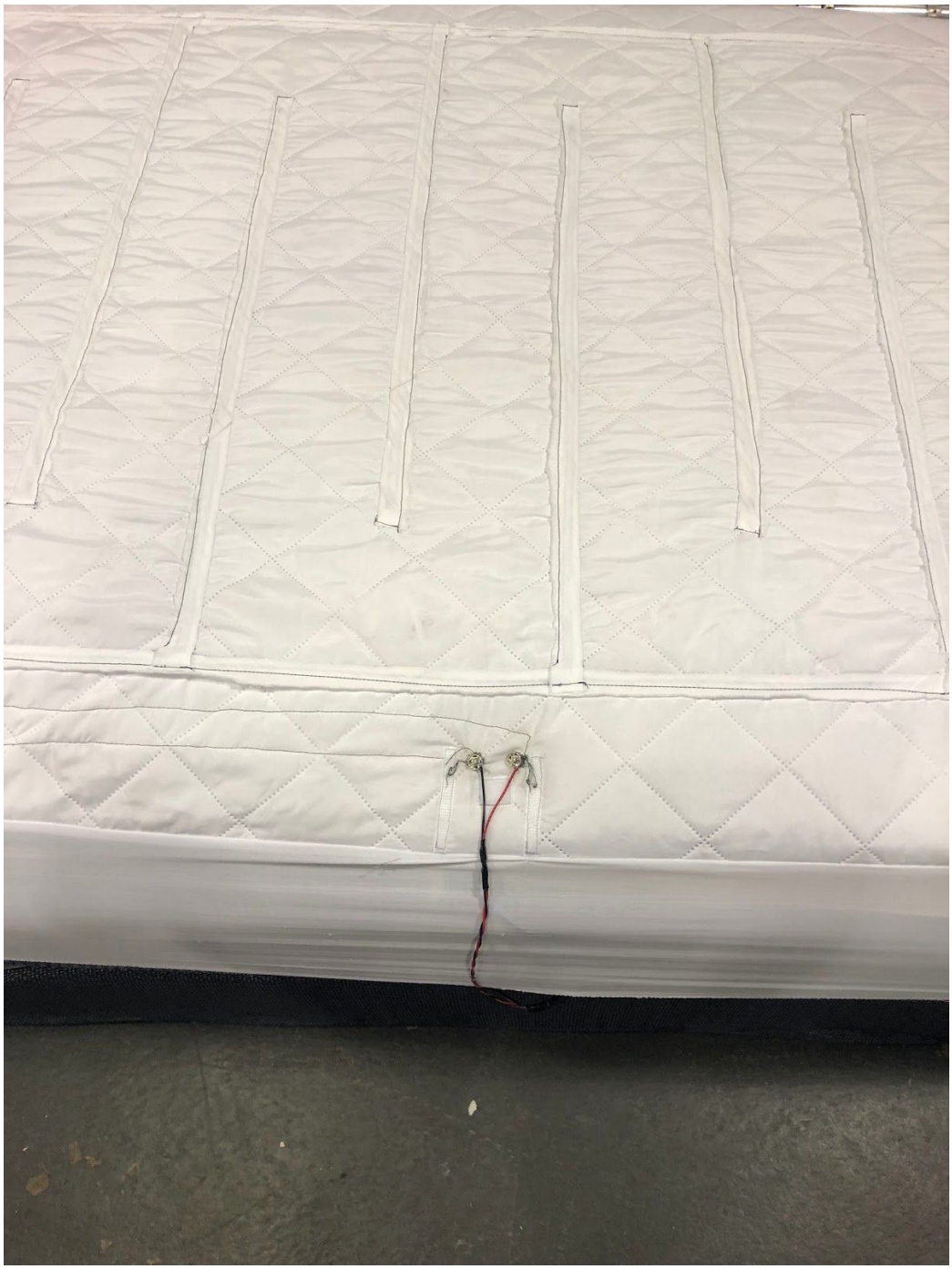


Figure A.1.10: Wires Snapped In Before being Sewn into Covering Patch

Figure A.1.11: Full Outside Design of Mattress Topper. Full design includes conductive thread wires snapped in, load cells underneath, and accelerometer between foam layers.

Conductive Thread Design

20 mL of 95 F water were poured in gap widths of 1 inch increments to decide on a reasonable gap-width between the two wires, as shown in Figure A.1.12.

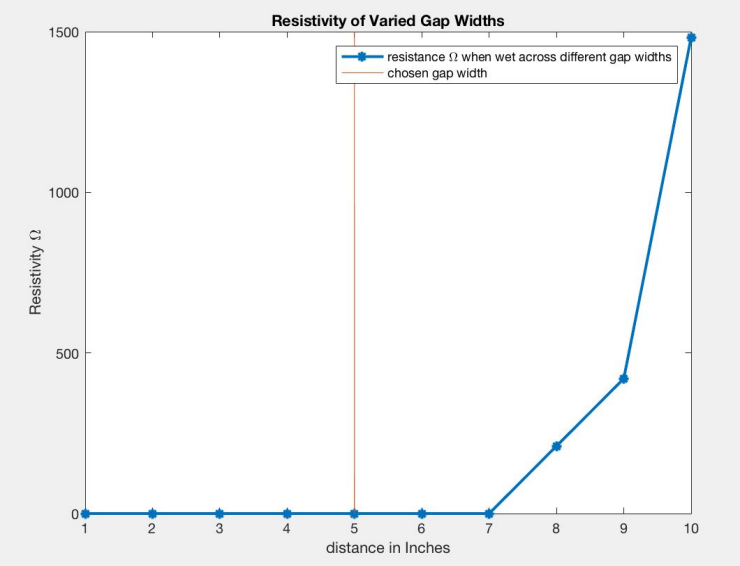


Figure A.1.12: Resistivity of Varying Gap Widths

We moved forward with 5 inches between each thread. This gave a good connection between threads when liquid was poured and also balanced the cost of adding too much thread.

20 mL of 95F water was poured in each of the circles shown on Figure A.1.13 and let dry to measure the resistivity over time when connected to the Arduino. The resistivity over time is as shown in Figure A.1.14.

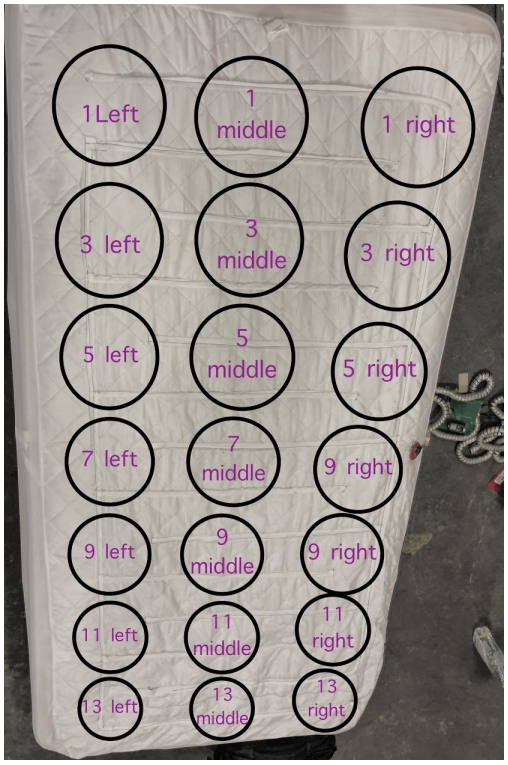


Figure A.1.13: Mattress Topper Wet Test Locations

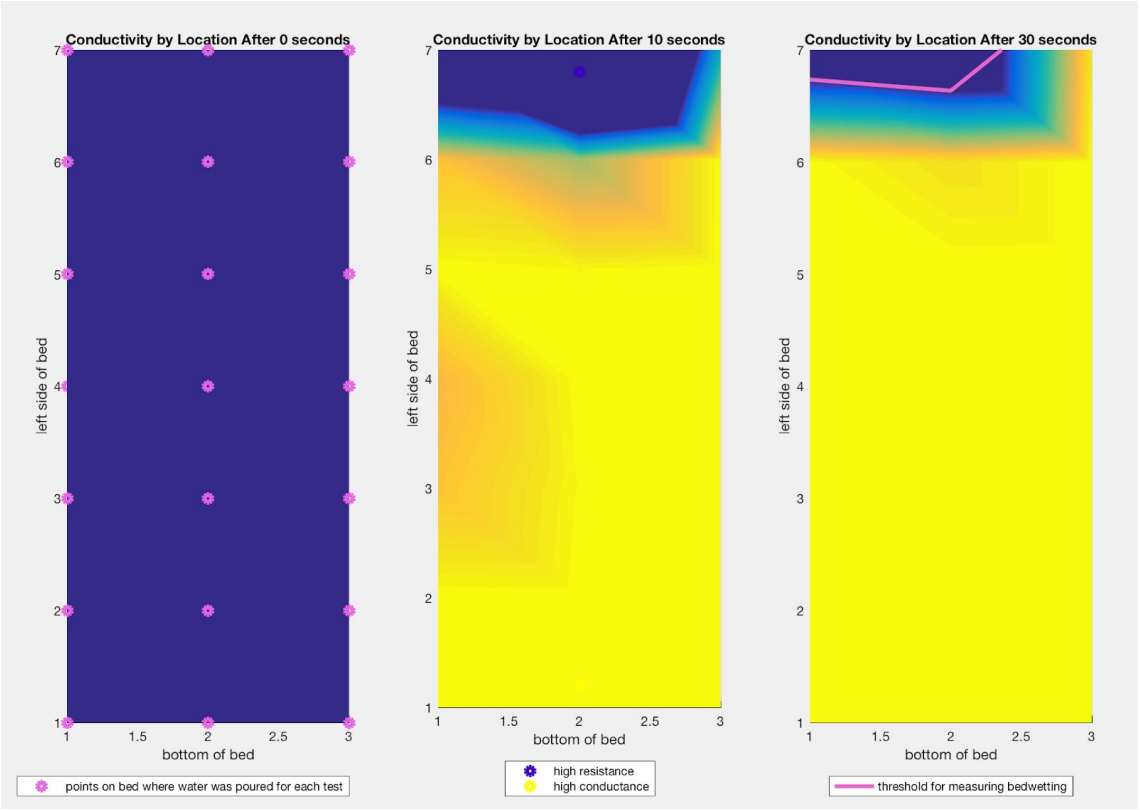


Figure A.1.14: Bedwetting Test Resistivity

The upper left corner of the bed (furthest from the connection to the Arduino) showed only a 30% drop in resistivity from dry. The rest of the bed (96.83%) dropped from a range of 1-2 Million Ohms of resistivity to a range of 0-20 kOhms.

We set a threshold of 500,000 ohms for detecting bedwetting. This threshold can be seen by the pink line on the resistivity map. This threshold was low enough to never trigger a ‘wet’ signal from signal noise when dry.

To test the durability of the conductive thread in the washer, the mattress topper was washed on cold and dried on delicate 15 times. After each wash the resistivity was tested while dry as shown in Figure A.1.15. After, 20 mL of 95F water were poured in the middle. The resistivity did not change significantly when wet or dry after each wash. The only factor for dry resistivity was the warmth (time after removal from dryer). However, this was still several orders of magnitude larger than the resistivity when wet.

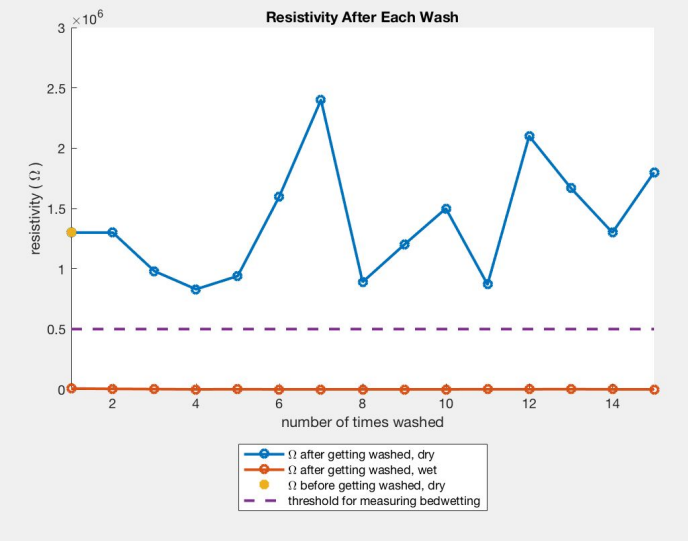


Figure A.1.15: Post-Washing Test Resistivity

Each of the two conductive thread lines terminates in a conductive metal snap (size 10 mm). This worked effectively to connect the electronics to the mattress topper with an easy set-up and a durable connection. The top of the metal snap was soldered to a wire that connected to the electronics.

Appendix 2: App Version Iteration and User Testing

Over the course of ENGS 90, we have conducted 60 user tests (1 client tester, 17 parent testers, 40 student testers, 2 Dartmouth Applied Learning and Innovation Lab Lead Designer testers) which we have used to inform our app design over the course of 7 versions. Summaries of remaining recommendations from user tests can be viewed in the next section. Details about each test and version update are included below. Only major changes between each version are discussed below.

V0:
Our first functioning version displayed only weekly metrics in a primitive bar chart and a settings page.

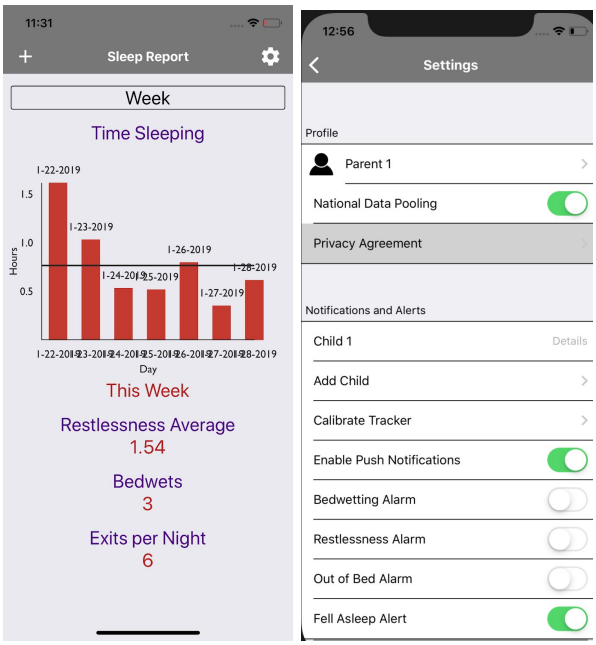


Figure A.2.1: V0

V1:
Our first version cleaned up the weekly screen graphics with new labels, added a line marking the average sleep for the week, and provided both a new daily details and full details page.

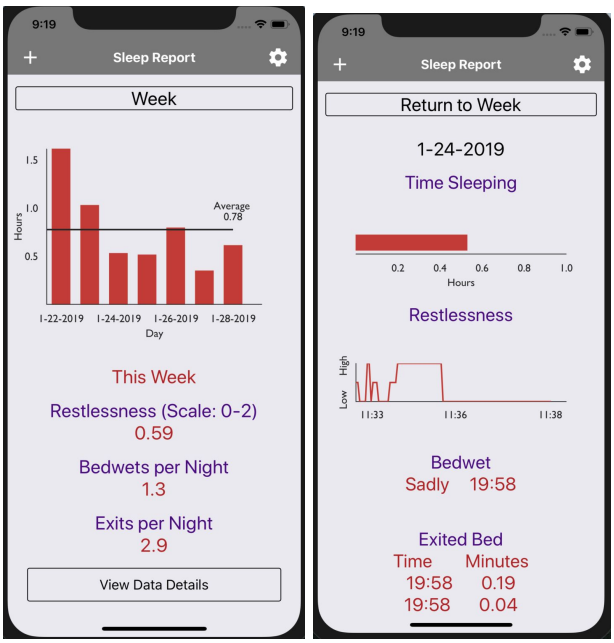


Figure A.2.2: V1

This version was evaluated by 10 user tests. Tests were conducted through online parent surveys showing screenshots of different app views with descriptions of functionality. Test results indicated that the users wanted a manner of viewing the time the child spent in bed, not just the time asleep. Users also wanted more clarity about what data metrics measured and to see labels for days of the week (Sunday, Monday, etc.) instead of only dates.

V2:
Our second version replaced the average sleep line with a line showing the total time in bed for each night. We also added info buttons to clarify the different metrics displayed. We also altered the sleep graph on the daily view to show time in and out of bed instead of total sleep duration.

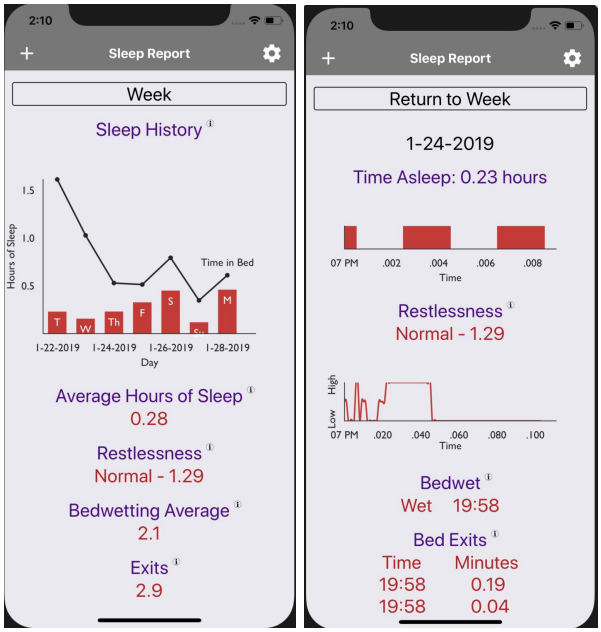


Figure A.2.3: V2

These changes were evaluated by 5 user tests. Testers indicated that they wanted an introductory screen or tutorial, a way to navigate to the Day screen by clicking on a bar under the weekly view, scrollable weekly views, and a way to view data on different time scales such as by month.

V3:
Our third version added a tutorial screen, implemented scrollable functionality for the day/week/month views, added a ticker to switch between day/week/month/all views, changed the color scheme, changed the date labeling, displayed times in hr:mm instead of decimal hours, displayed military time in AM and PM format, and changed the averages list to a 2 column layout.

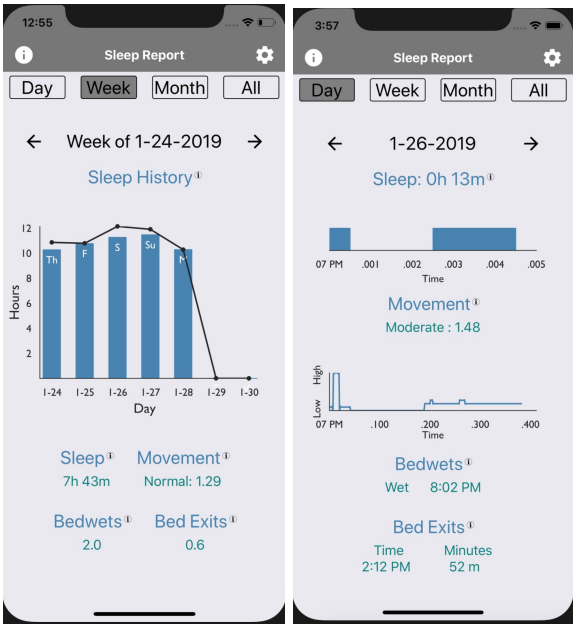


Figure A.2.4: V3

These changes were evaluated by 5 user testers. Users indicated that the black line showing time in bed was confusing, the axes labels on the daily view were unclear, they did not like having the national averages comparison on the "All" page, and they wanted further data clarity with units and shorter descriptions of metrics.

V4:
Our fourth version added a buffer to the bottom of all screens because users were having trouble scrolling down, added a message info box when the black in-bed line was clicked to clarify its meaning, fixed the time scales on the daily view, moved national graph comparisons to a different page, and reworked the metric descriptions to be more clear.

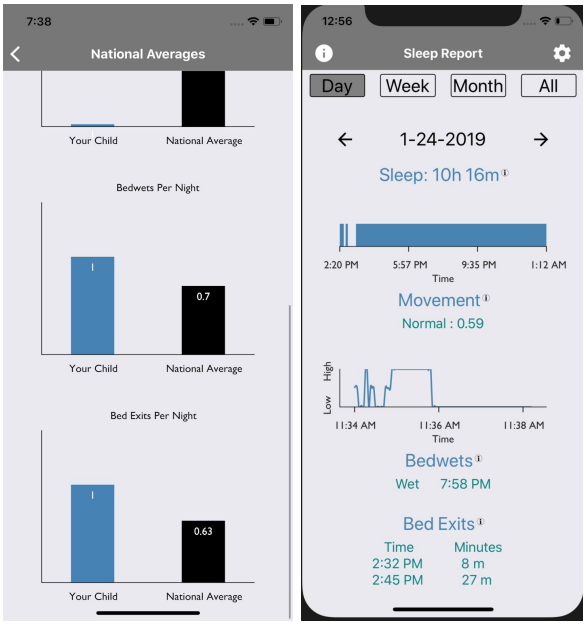


Figure A.2.5: V4

These changes were evaluated by 5 user testers. Users indicated that they wanted to see bedtimes and waketimes in addition to sleep durations, viewing 0 minute bed exits was unhelpful, the movement line graph on the daily view was confusing, and that the visual layout could be improved.

V5:
Our fifth version more clearly delineated when the user had reached the end of the scrolling range of dates, removed 0 minute exits from the display, added A/B/C testing for 3 graphs, and began to show color scheme changes. Our A/B/C graphs tested 3 sleep views: the original line/bar view, a stacked bar showing time asleep and time awake for each night, and a floating bar showing the bedtime and waketime.

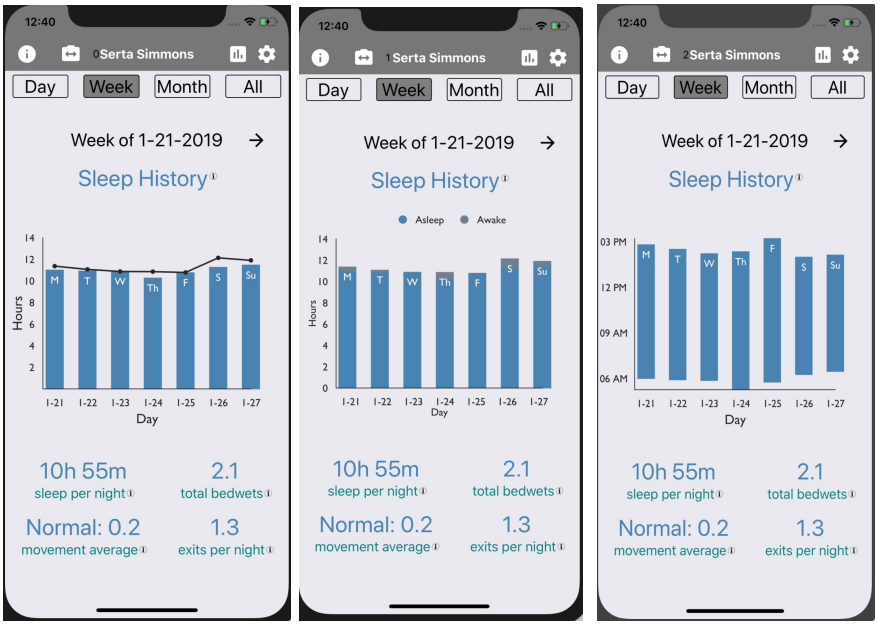


Figure A.2.6: V5

These changes were evaluated by 10 user testers. Users indicated that the navigation to the day screen by clicking on a bar in another view was too abrupt, the colors were still not visually appealing, and they preferred the stacked bar view over the bar/line view. Users also indicated that the floating bar view showing bedtimes and waketimes showed new information that they would be interested in seeing in addition to the total sleep duration.

V6:
Our sixth version fully implemented the sleep graph toggle view on the week and month views so that users could switch between seeing a stacked bar chart showing awake, asleep, and nap time, and a bar chart showing nightly bedtime to waketime. This version put restlessness on an easier to understand 0-100 scale, changed the axis labels again, totally reworked the visual graphics, added metric trend charts to the All view, added restlessness considerations into our sleep algorithm, and altered the navigation between summary views and the day view. Instead of clicking bars to see the daily view, users now clicked bars to see summary details. By clicking the details, they were transported to the day page.

Figure A.2.7: V6

These changes were evaluated by 24 user testers. Users indicated that they were generally satisfied with the app design & navigation as well as the data complexity & visualization. Users also saw high potential marketability of our sleeptracker for children. The most confusion centered on understanding the movement graphs and averages. More detail about this confusion is discussed under the Recommendations section. As future features also discussed in the Recommendations section, parents also want to see the numerical durations associated with asleep, awake, and napping in the sleep graph. Parents would like to see the daily details as a pop-up callbox or other visualization instead of the daily details button currently located under the blue "Sleep" heading. Clicking on this new callbox/visualization should navigate parents to the daily view page.

Appendix 3: Individual User Tests by App Version

V0 and V1 Testing:
Parent Testers 1-10:

- Want to see time in bed, not just time asleep
- Want to see a mix of dates and days of week on different graph views
- Want to see a variety of time ranges, not just 1 week
- Want to see times of bedwets, not just number of bedwets
- Want national averages over community averages
- Want a daily view in addition to a weekly view
- Want a way to differentiate normal sleep from restless sleep

V2 Testing:
Student Tester 1

- "What do I do?"
- Animation too slow
    - Got confused and kept clicking while animation was loading
- "Does it only show a week per time?"
    - Wants to scroll on weekly summary
- "As a first time user, I don't know what it was for. There was no intro, no tutorial. There should be just a one time tutorial that you click on once"
- "It's interesting that the bar graph itself was like ...you were trying to show a trend, right? Maybe show between what times...between 8 and 6, or 9 and 6. It makes a difference...it could make a difference about when I send my kid to bed"

Student Tester 2

- App is not intuitive
    - "If I click on week, what am I going to?"
- What about naps?
- "Should Add Child be more difficult to access?"
    - Add Child should be moved out of header into Settings
- "Do multiple parents use the same app for the same data?"
    - Can two parents get the same data?
- "What's the dot versus the bar graph?"
    - Axes are misleading
- "Too much text"
    - Maybe vary font size and text, or change color
    - Look at sleep psychology to pick colors

Student Testers 3&4

- Metrics need units
    - Bedwetting average is 2.1 what?
    - What is restlessness?
- It might make more sense if the data was right. But what is the difference between the data I can get on the daily page and the averages?
- Can you customize the metrics shown? I might not care about bedwetting but might want to track bedtimes, etc.
    - Can you show the average bedtime?
- Consider difference between crib and bed sleepers, with and without railing (not leaving bed, but maybe jumping around)
- Parents might be nervous about putting sensors in bed

Student Tester 5

- Tutorial description on weekly needs to say to scroll through this page first, then click on day
- Get rid of minus sign under restlessness, and make it a colon
- "Oh this is great" → regarding daily view
- What units are these in?
- Can I see AM or PM instead of military time?
- Axes are still messed up

V3 Testing:
Student Tester 6

- Having to scroll up to see bottom of screen → We should add padding to bottom of screen
- Thinks hours of sleep is intuitive
- What are the units? Per night?
- Likes national averages
- Prefers seeing sleep duration over bedtimes and waketimes

Student Tester 7

- What are the units?
- Description on daily sleep graph is a little confusing
    - Maybe could use a clock or circle instead of a bar graph
- What is the movement metric?
- Bedwetting is intuitive
- Likes summary statistics
    - May be nice to see at what points of day the kids were asleep for, instead of total number of hours
    - Would like to see this on weekly summary

Student Tester 8

- What's in the sleep report?
- Thinks the info buttons give helpful descriptions
- Wants more statistical figures (like bedwets per night) instead of long descriptions
- The tutorial should say what the bar vs line mean in the sleep graph
    - If the user clicks the line, a label can pop up

DALI Lab Testers 1-2

- Consulted for color scheme
- Consulted for app navigation ideas
- Consulted for text layout

V4 Testing:

Student Tester 9

- What's the scale on restlessness?
    - The info button is not as helpful as she wants
- Lot of zero minute exits
- Not clear that pressing i toggles on and off the tutorial mode
- Need more padding at the bottom of the screen
- Maybe you shouldn't average naps into total hours slept

Student Tester 10

- Graph makes sense, averages make sense
- Not sure what movement average means
- What is the total bedwets period for? (Specify whether for month or week or...)
- What do exits mean?

Student Tester 11

- I thought the arrow would bring me back to the week screen instead of scrolling through days
    - Back button between day and week views?
- Axis gets messed up when there's only zeros for the week
    - Check to make sure this error case is handled
- Info description is helpful on bar vs line on weekly sleep graph
- Likes averages and likes bar for recommended sleep on national averages

Student Tester 12

- Make it more colorful and change the fonts
- App is intuitive, but it needs a little more fun

Student Tester 13

- I like the toggles and how clicking on day brings you straight to day
- Movement graph is confusing -- What do high and low mean?
- Wants to hover over bars to see summary statistics (maybe double tap to get full day view)
- Should catch NaN error

Client Tester 1:

- Client is relatively happy with overall design, metrics, and navigation
- No major comments except regarding small bugs fixed before Version 5

V5 Testing:

DALI Lab Tester 1:

- Concerning main graph 2 on week, show the total hours of sleep instead of the “M,T,W..” and show the “M,T,W…” on the axis instead of the “1-29 1-30 …”
- Unclear that titles are clickable → info button is tiny by them
- New name for ‘Averages’ e.g. “statistics”, “how you stack up”, “comparisons”
- Likes the stacked bar view over the bar/line sleep graph view

Student Tester 14

- Day: What does 1-25 mean?
- Not clear that time in bed vs asleep is different
    - What are exits? What does that phrasing mean?
- I like the main graph (easy to understand)
    - I like the layout of the main week page
- App needs new colors!
- She likes the bar and line over the stacked bar view

Student Tester 15

- Clicks on day by accident and goes to day screen → Doesn't want navigation to happen like that
- Movement info isn’t very helpful
- Summary graph needs a little more clarity
    - Time spent in bed, not time in bed → Change wording
    - How can you have 2.1 bedwets? → Clarify averages
- Don’t need info button for bedwetting and exits (intuitive)
- Likes national averages
- Tutorial is a lot of words → shorten
- Need tap to see details of the day instead of immediately navigating you to Day view
- Should have a feature to log data if you slept in a different bed

Parent Tester 11

- “I love that” about sleep vs in bed on sleep graph
- So what are the times under there? → Confused about bed exits on daily view
- “I have not had a child that had trouble sleeping for a long time, but I can imagine this would be cool. To see the sleep history...Certainly when you have a bedwetter, it’s helpful to know because you can see what’s triggering it. And just understanding their sleep per night is pretty cool because every kid is different”
- Liked tutorial text
- Wants trends of exits → Add trend data visualizations

Student Tester 16

- Axis label on floating graph of bedtimes/waketimes is confusing
- Maybe leave in the toggle to go between offset and the stacked/bar instead of choosing only one option after A/B/C testing

Student Tester 17

- Need type check for numbers on user input boxes
- Getting NaN errors
- Confusion about going from week to day
    - Doesn't think there's enough of a UI cue to know this feature
    - A little bit jarring to him
- Would like to be able to swipe to switch views instead of using the arrows
- Need to get rid of bar graph labels if there's no data
- I don’t like the floating bar view → it’s hard to get a glance and know what’s up
- Wants more details on the averages/info

Student Testers 18-21

- Preferred stacked bar over bar/line sleep graph, bringing the totals of A/B/C testing to:
    - Bar/Line Sleep Graph: 1
    - Stacked Awake/Asleep Sleep Graph: 10
    - Floating Bedtime/Waketime Graph: 0 (5 found interesting)

V6 Testing:

Student Testers 22-40 & Parent Testers 12-17:

- Conducted Use-Value tests, as detailed in Appendix 3
- Most parents liked the idea of the app
- The most confusion related to the meaning of movement averages/graphs
- Many parents also did not find the sleep graph toggle, but this may be solved with a tutorial
- Some parents could not see the info button on the metrics. SSB should consider making the button bigger in the next design iteration
- Parents want to see numbers associated with asleep, awake, and napping on the sleep graph. They should allow parents to see daily details by clicking on a bar on the week/month/all view

Use-Value Survey and Results:

Users were provided the Use-Value scale as a reference and given the following prompts:

Rank the visual aesthetics of the app
Rank the ease/intuitiveness of app navigation
Rank the clarity of app graphs and data
Rank the data complexity (Is the information detailed?)
Rank the potential product marketability

Answers were provided on a 0-10 scale, as detailed below.

Use - Value Analysis	
Solution Value	**Points**
absolutely useless	0
very inadequate	1
weak	2
tolerable	3
adequate	4
satisfactory	5
good, w/drawbacks	6
good	7
very good	8
exceeds requirements	9
excellent	10

Figure A.3.1: Use-Value Scale

Version	Tester	Visual Aesthetics	Ease of Use	Data Clarity	Data Complexity	Marketability
V4	Student	7.5	10	10	9	10
	Student	6	8	6	9	8
	Student	5	7	10	8	9
	Student	6	6	7	7	8
	Student	7	6	8	7	7
V5	Parent	8	8	9	10	9
	Student	7	5	10	8	10
	Student	9	9	8	8	10
	Student	9	10	10	10	10
	Student	6	6	6	8	10
V6	Student	8	6	9	8	9
	Student	8	7	9	10	9
	Student	9	8	8	8	10
	Student	7	6	7	9	8
	Student	10	8	10	8	6
	Student	8	7	6	8	7
	Student	9	8	10	9	8
	Student	8	7	9	10	9
	Student	9	8	10	9	10
	Student	7	7	9	6	7
	Student	9	8	10	6	8
	Parent	6	8	6	7	7
	Parent	8	8	7	9	9
	Student	8	8	6	8	7
	Student	6	8	7	8	8
	Student	6	9	5	9	9
	Student	6	7	8	9	5
	Student	5	6	6	8	8
	Student	6	7	9	9	8
	Parent	10	10	10	10	8
	Student	4	8	8	8	5
	Parent	5	8	7	6	8
	Parent	9	9	10	10	10
	Parent	9	9	8	10	7
	Parent	9	3	6	10	8
	V6 Mean	7.56	7.52	8.00	8.48	7.92
	V6 Median	8.00	8.00	8.00	9.00	8.00

Appendix 4: Integrated System Tests

Comfort Test

Use-Value Survey and Results:

Users were provided the Use-Value scale as a reference and given the following prompt:

How comfortable was the bed?

Users were also asked:

Can you see our device?

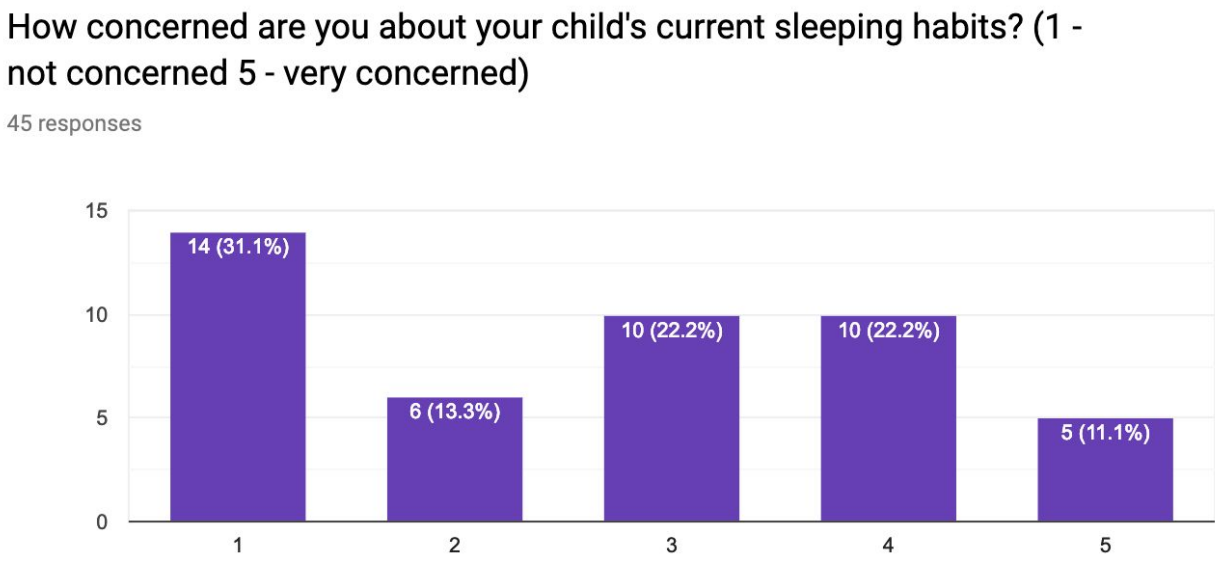
Can you feel our modifications? If so, where is the inconsistency in the mattress?

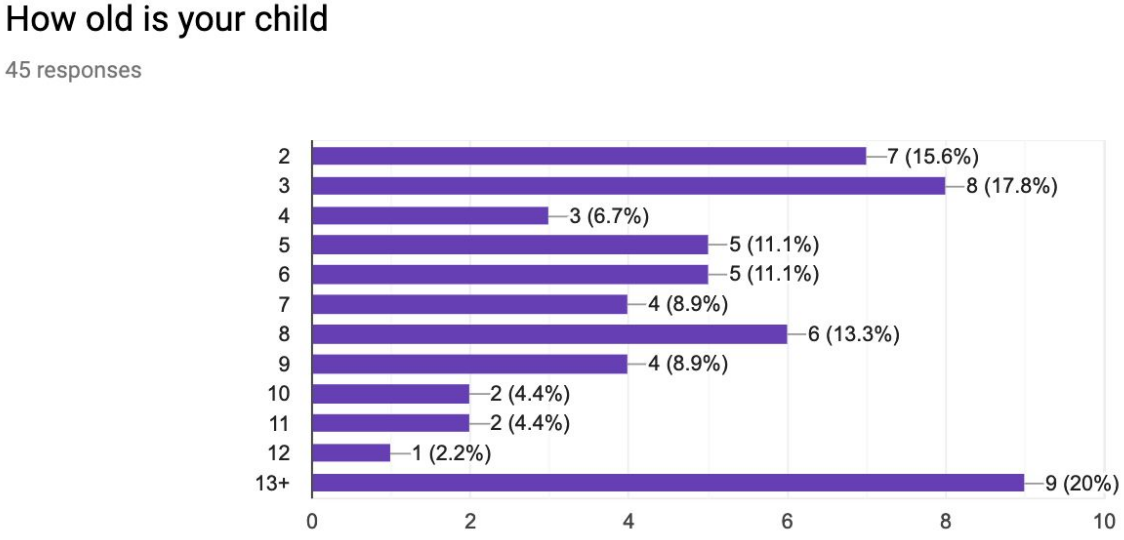
Answers were provided to the first prompt on a 0-10 scale, as detailed in Figure A.2.1. Answers to further questions were "Yes" (y) or "No" (n). An incorrect guess about the location of the mattress modification for the 2nd question is indicated by (w).

Subject Number	Comfortable Ranking	Can you see it?	Can you feel it?
1	8	y	n
2	8	n	n
3	8	n	n
4	8	n	n
5	8	n	n
6	7	n	w
7	8	n	n
8	9	n	n
9	7.5	n	n
10	7	n	n
11	8	n	n
12	8	n	n
13	10	n	n
14	10	n	n
15	9	n	n
16	8	n	w
17	8	n	w
18	9	n	n
19	10	n	n
20	8	n	n
21	10	n	n
Average score:	8.40		

Appendix 5: Market Research

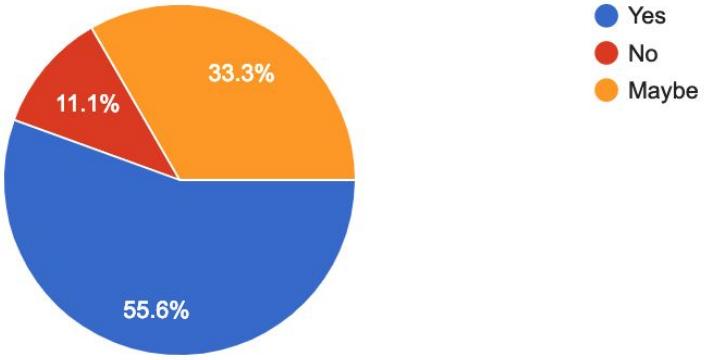
Survey results from 45 parents are included below:





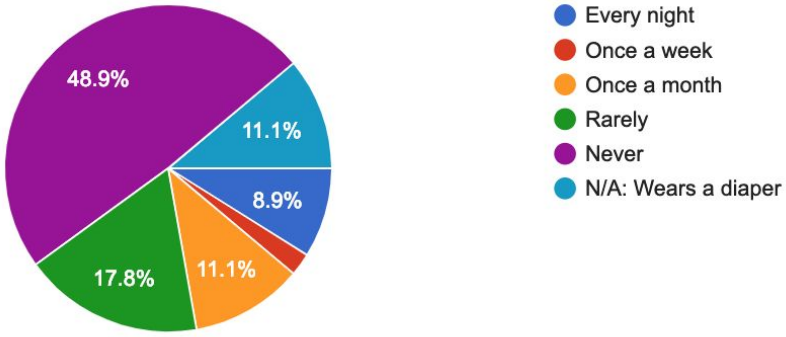
Are you interested in learning more about your child's sleep?

45 responses



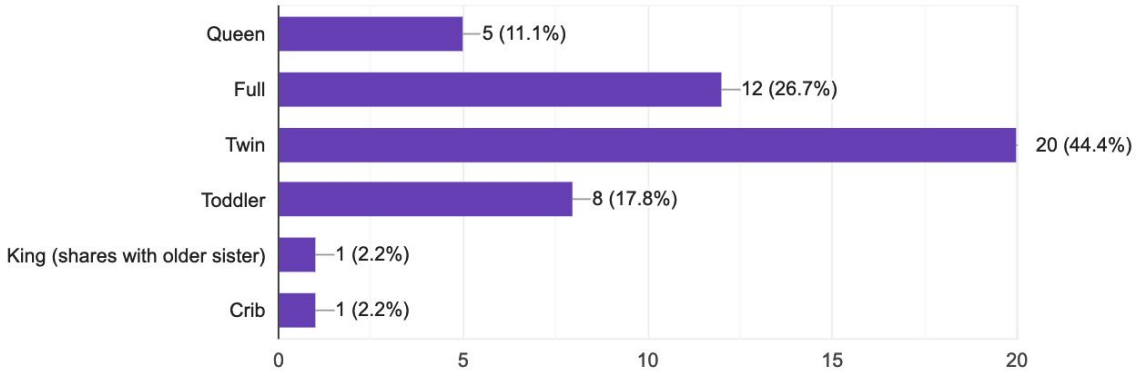
How often does your child wet their bed?

45 responses



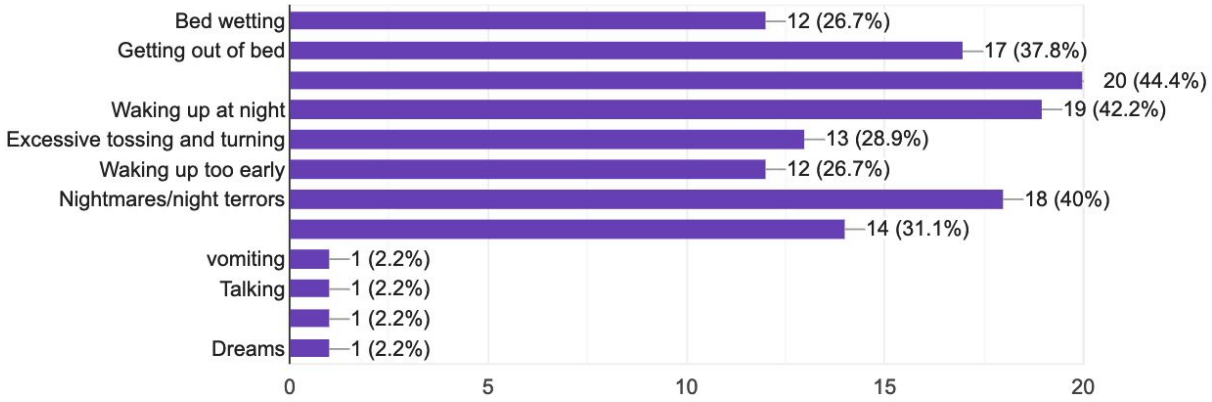
What size bed does your child have?

45 responses



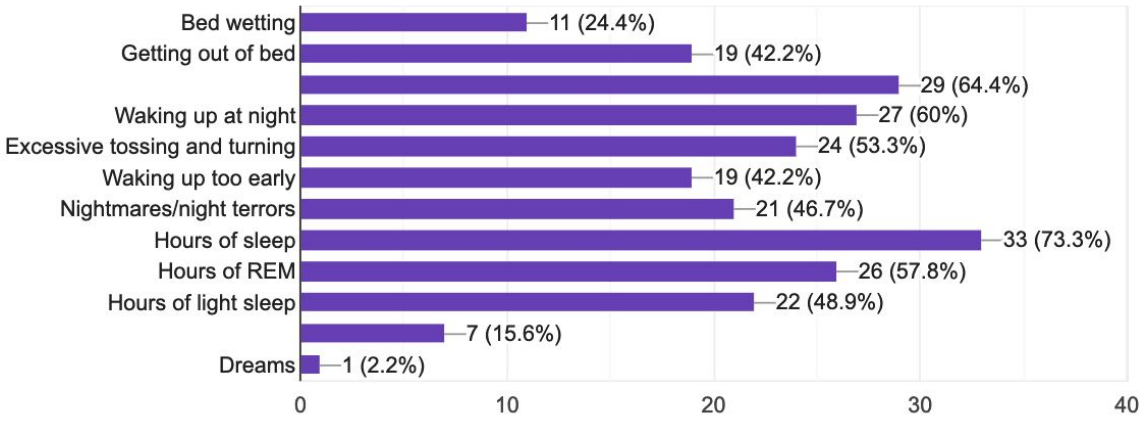
Would you be interested in real-time notifications about any of the following?

45 responses



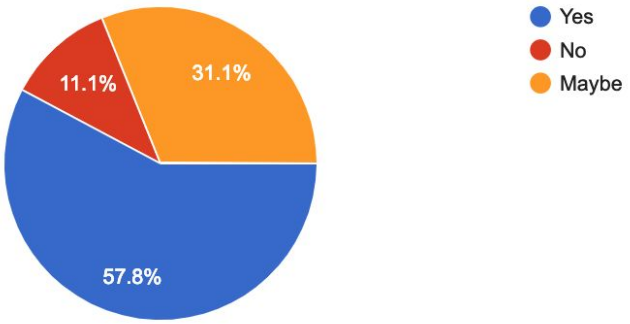
Would you be interested in weekly/monthly information summaries about any of the following?

45 responses



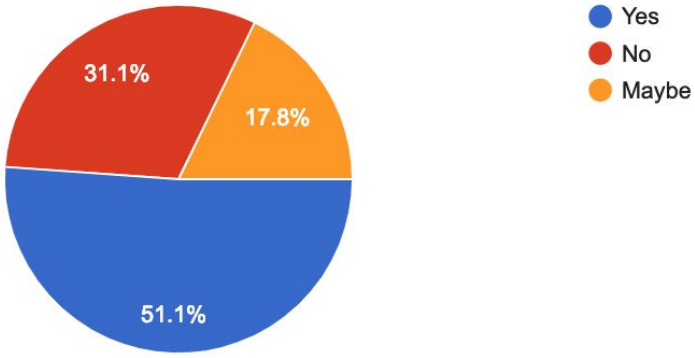
Are you interested in also tracking your child's mood to compare with his/her sleeping?

45 responses



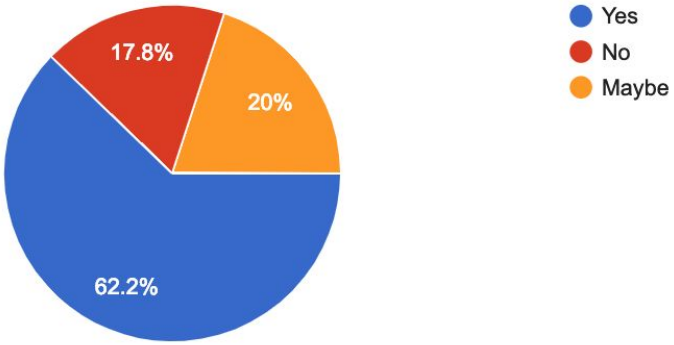
Are you interested in comparing your child's data to community data?

45 responses



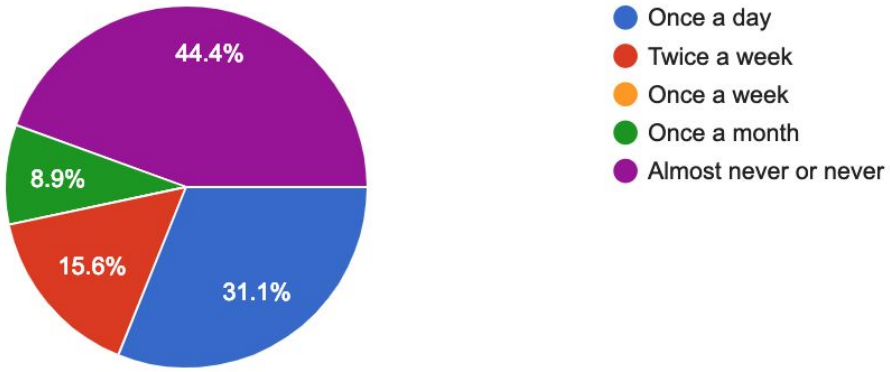
Are you interested in comparing your child's data to National Institute of Health recommendations?

45 responses



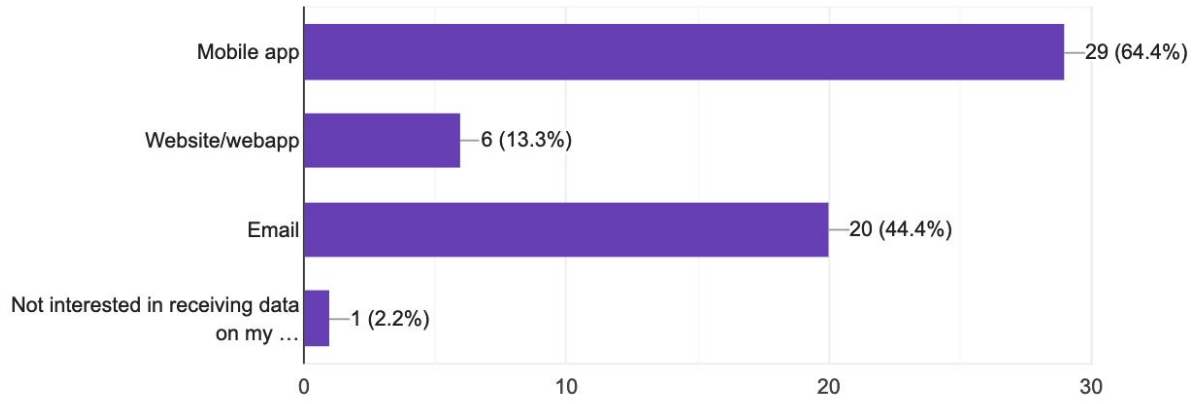
How often does your child nap?

45 responses



How would you like to view information about your child's sleep?

45 responses



Individual survey results can be viewed at: <https://docs.google.com/spreadsheets/d/1teeF0XLE0gMpgwu37xo7Q3Rdaqgb2PTNez0snm6uxg0/edit?usp=sharing>